Lazy FCA report

1. Dataset and introduction:

I choose the Titanic dataset, which provides detailed information about passengers during the shipwreck. The binary classification is based on the fact whether the passenger survived the crash or not.

In my project, I binarized the variables of the selected dataset, trained the Lazy FCA algorithm on this data, and compared the metrics on the deferred sample with the metrics of classical classification algorithms.

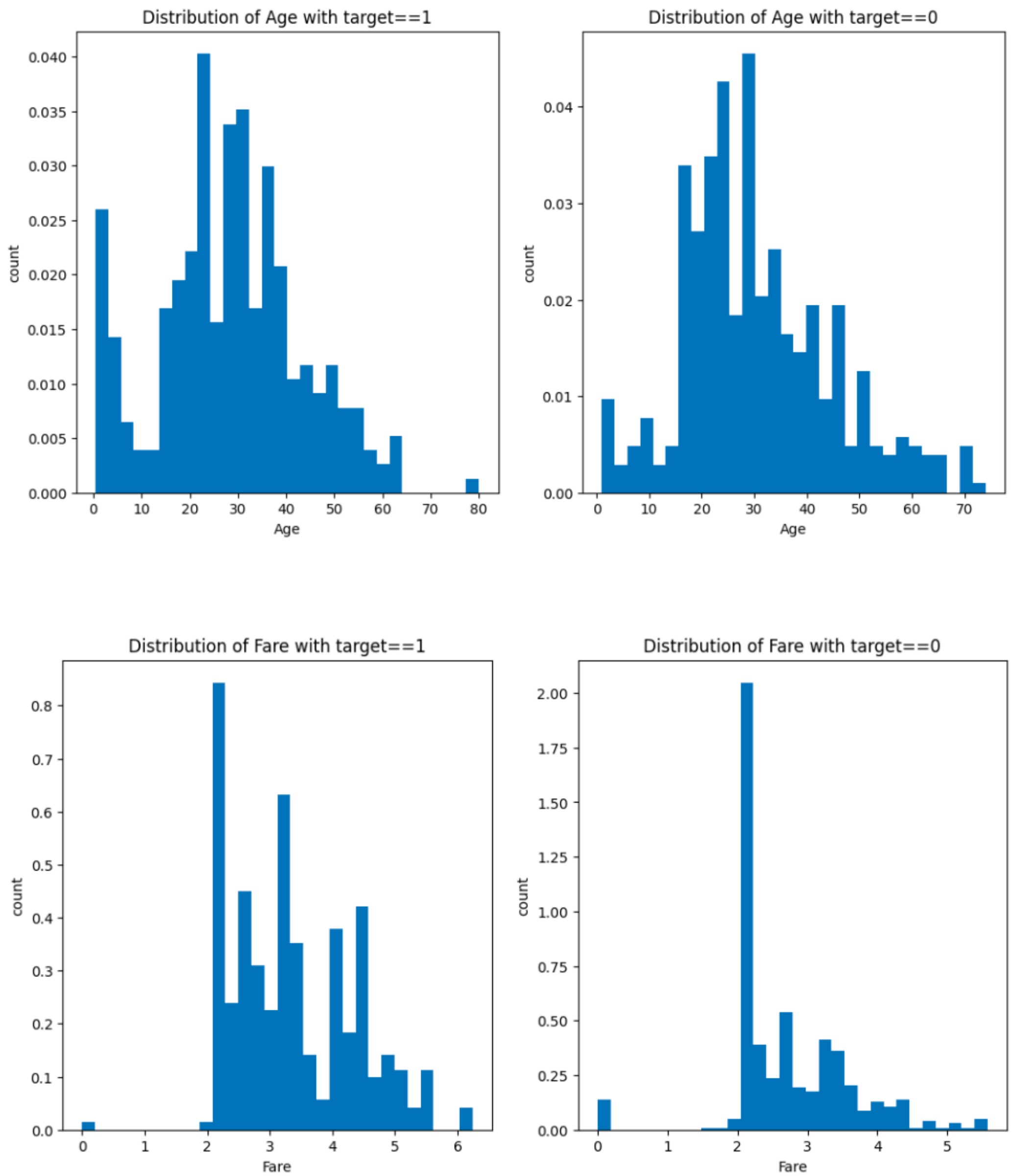
The dataset provides data such as the age and gender of the passenger, as well as information on the price, class and location of the seat according to the ticket.

2. Features binarization:

I have divided all available features into 3 categories, binary variables (Sex), categorical variables with more than 3 unique values (Pclass, SibSp, Parch, Cabin, Embedded) and numeric variables (Age, Fare).

Accordingly, I applied dichotomic scaling to binary variables, nominal scaling to categorical variables, and ordinal scaling to real variables (in addition, I applied a logarithm to the Fare variable to get a distribution closer to normal), after which I visually selected thresholds for binarization

These are the distributions of Age and the logarithm of Fare, respectively



3. Aplicaion of Lazy FCA algorithm:

I divided the dataset into train and test parts (20% of all data got into the test part), wrote functions to calculate all the required metrics and made predictions using Lazy FCA algorithms.

4. Comparison with other classification algorithms:

I used 5 other standard classification methods: KNN, Logistic Regression, Decision

Tree, Random Forest, XGBoost, and here are the results obtained in comparison (the data for training these models are the same, but without binarization, one hot encoding was done for categorical variables and standard scaling for real ones):

	True Positive	True Negative	False Positive	False Negative	Negative Predictive Value	False Positive Rate	False Discovery Rate	accuracy	precision	recall	f1
fca	41	98	12	28	0.777778	0.109091	0.405797	0.776536	0.773585	0.594203	0.672131
knn	46	94	16	23	0.803419	0.145455	0.333333	0.782123	0.741935	0.666667	0.702290
decision tree	47	93	17	22	0.808696	0.154545	0.318841	0.782123	0.734375	0.681159	0.706767
logistic regression	48	98	12	21	0.823529	0.109091	0.304348	0.815642	0.800000	0.695652	0.744186
random forest	50	97	13	19	0.836207	0.118182	0.275362	0.821229	0.793651	0.724638	0.757576
xgboost	45	97	13	24	0.801653	0.118182	0.347826	0.793296	0.775862	0.652174	0.708661

Let's go through the main metrics responsible for the quality of classification:

For the FCA algorithm, it turns out that accuracy and $f1$ measure are the lowest of the presented algorithms, but are quite close to the basic KNN and Decision Tree algorithms.

Basically, the drawdown occurs due to the fact that the model confidently defines class 1 (precision), better than many models, but at the same time it very poorly highlights all elements of class 1 (a large value of False Negative) (recall). When analyzing this problem, it was highlighted that only 50% of all examples receive at least one non-zero number of classifiers (positive or negative). Accordingly, due to a small imbalance of classes in the dataset, it is difficult to choose a threshold for the number of classifiers and it is better to assign all undefined samples to class 0, which is more.

Random Forest and Logistic Regression performed best on the considered task, with accuracy 5 percentage points higher.